WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, September 06, 2004

Hide?	Set Name	Query	Hit Count
	DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ		
☐	L4	19991110	1
☐	L3	L2 and (remote or gateway)	11
☐	L2	L1 and network	14
☐	L1	request near8 (convert or transform or converting or transforming) near8 (higher adj3 level)	14

END OF SEARCH HISTORY

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L4: Entry 1 of 1 File: USPT Jul 12, 1994

DOCUMENT-IDENTIFIER: US 5329619 A
**** See image for Certificate of Correction ****
TITLE: Cooperative processing interface and communication broker for heterogeneous computing environments

Abstract Text (1):
An object interface is disclosed that supports three modes of inter-object communication--message processing (store and forward), conversational communication, and remote procedure call. A service broker manages service requests from, and responsive services provided by, a plurality of clients and servers, respectively, which may reside on different hardware platforms and operating systems and may be connected to computer networks having different network architectures and associated communications protocols. The broker manages the service offerings from servers and service requests from clients, and clients and servers communicate and exchange information with one another via the broker. The service broker includes different application programming interfaces for allowing participants to access the functionality of the service broker.

Application Filing Date (1):
19921030

Brief Summary Text (2):
The invention relates to computer networks, and specifically to a service broker for clients and servers operating in a heterogeneous computing environment.

Brief Summary Text (3):
It is increasingly common to interconnect multiple computing systems into a computer network, such as a local area network ("LAN") or wide area network ("WAN"). In a computer network, a number of computers are joined together to exchange information and share resources. A network is a distributed computing environment in which networked computers provide users with the capabilities of access to distributed resources, such as remote files and databases or printers and of distributed processing, in which an application may be processed on two or more computing systems. In such a distributed computing environment, a computing application's component functions may reside on different machines but work together. For example, each work station or personal computer ("PC") in the network often provides user interface processing and local analysis and processing, while larger, host computers may maintain a large set of files and coordinate access to large databases.

Brief Summary Text (4):
In the distributed computing environment, each application must be able to communicate and exchange information with other applications or machines in the environment. If all the machines are based on the same hardware platform, use the same operating system, and are interconnected using a single network architecture and communication protocol, connection and communication between applications and/or machines is straightforward. However, this ideal is seldom achieved. There are many different (and often mutually incompatible) computer network architectures (such as SNA, OSI, TCP/IP, DECnet, and LANs), hardware platforms (such as IBM, DEC,

WANG, and Apple), operating systems (such as UNIX, OS/2, MS-DOS, VMS, and MVS), and application languages (such as COBOL, FORTRAN, PL1, C, and NATURAL). This heterogeneity presents an obstacle to the connectivity and interoperability of the systems.

Brief Summary Text (6):
The system of FIG. 1 is heterogeneous because each of the clients 4 and servers 6 may be applications running on different hardware platforms under different operating systems, and/or within different computer networks. For example, the computer for client 4a may be an IBM mid-range computer running the OS/400 operating system, the computer for client 4b may be an IBM PC running MS-DOS or OS/2, while the computer for server 6a may be a UNIX-based work station, the computer for server 6b might be a DEC mainframe computer, and the computer for server 6c might be work station running IBM's OS/2. Other computing systems might also be connected, such as a work station running Microsoft Windows or an Apple Macintosh. LAN 5 might be based on IBM's System Network Architecture ("SNA") and IBM's Logical Unit 6.2 ("LU 6.2") communications protocol, while LAN 7 might be based on a different architecture, such as OSI and its associated communications protocol. The communications protocol is a defined set of procedural rules which computers use to communicate across a network.

Brief Summary Text (7):
The use of different hardware platforms, operating systems, or network architectures and their associated communications protocols inhibits the useful exchange of information between clients and servers in a heterogeneous environment, such as that shown in FIG. 1.

Brief Summary Text (8):
Effective operation of the heterogeneous computing environment of FIG. 1, with its different and incompatible hardware, software, and network architectures, requires some mechanism for matching service requests from clients with the appropriate service offerings from servers and for managing the communications between clients and servers.

Brief Summary Text (9):
Developers of software applications that would be used in client-server relationships face three problems: software portability (platform independence); network transparency); and reliable data delivery (store-and-forward operation). These problems have not yet been adequately addressed. Simply providing a common interface, such as the Transmission Control Protocol/Internet Protocol (TCP/IP), only addresses part of the problem. Rather, the correct functionality must be provided to client-server application developers to allow applications to be developed on a "logical" platform with transparent communication across networks and with reliable data delivery.

Brief Summary Text (11):
U.S. Pat. No. 4,604,686 to Reiter, et al. ("Reiter") discloses an interface for inteconnecting user terminals 12 and diverse processors running different types of databases 14. The interface is a file-driven computer program for controlling access to the many databases by the user terminals. The interface is loaded with files having information relating to interfaces used with different processors, query languages, and data base managers, and information on the location of each kind of on formation and method of retrieval. A user at one of the user terminals makes a request for specific information to the interface. The interface in turn couples to each required data base (mimicking an asynchronous terminal), retrieving the data. Reiter provides remote access to heterogeneous computer systems in the form of asynchronous terminal emulation. Reiter controls data retrieval from these systems by a user-written command procedure, and presents the data in specific formats on the user terminal, again controlled by user-written command procedures. provides data retrieval from these systems. Reiter thus describes a very specific

means to automate dial-up, logon, data access, and screen formatting procedures.

Brief Summary Text (12):
There is thus a need for a system that facilitates cooperative processing among application programs in a heterogeneous computing environment and that provides store-and-forward messaging, conversational program-to-program communication and remote procedure calls. Such a system should support communication between applications independent of operating system, hardware, network/communication protocol, and programming language.

Brief Summary Text (14):
The drawbacks of the prior art are overcome by the method and apparatus of the invention. An object interface is provided that supports three modes of inter-object communication--message processing (store and forward), conversational communication, and remote procedure call. A service broker manages service requests from, and responsive services provided by, a plurality of clients and servers, respectively, which may reside on different hardware platforms and operating systems and may be connected to computer networks having different network architectures and associated communications protocols. The broker manages the service offerings from servers and service requests from clients, and clients and servers communicate and exchange information with one another via the broker. The service broker includes different application programming interfaces for allowing participants to access the functionality of the service broker.

Brief Summary Text (15):
An adapter may also be provided as a gateway to convert a foreign communications protocol to the function server protocol to allow applications programs to access the service broker functionality even though they are not compatible with the application program interface and function server protocol of the invention. The broker also provides services such as directory and naming services, message queuing, and accounting. The directory service includes a list of all available service offerings, while the naming service maps the logical name of each server and client to real addresses to allow the broker to match client service requests with actual service offering by servers. The message queuing service allows for store and forward type asynchronous request processing in which the client need not wait for the server to respond before continuing processing.

Drawing Description Text (6):
FIG. 4 is a schematic illustration of the use of stub programs in a remote procedure call.

Detailed Description Text (3):
FIG. 2 illustrates a distributed computing system including clients 10a-10d, which are interconnected to servers 12a-12d via a communications network 22 and service broker 14. Each of the clients 10 and servers 12 may operate as a client or a server, depending on whether it is requesting or supplying services. In this example, clients 10a and 10b are nodes of a first LAN, while clients 10c and 10d are nodes of a second LAN. Similarly, servers 12a and 12b, and 12c and 12d are nodes of third and fourth LANs, respectively. Each of the four LANs are based on a different network architecture, utilizing a different communications protocol, while each of clients 10 and servers 12 run different operating systems.

Detailed Description Text (5):
Thus, to request a service function, a client issues a call to a subroutine. The subroutine then establishes the link to the local or remote server via the service broker. Client requests can be execute synchronously or asynchronously. In the synchronous request, the client process waits for processing of the service before it continues with its own processing. A typical example of synchronous request processing is a data base request. When a client sends a request to a data base server, its own process is stopped until the server delivers the data.

Detailed Description Text (9):
A server is a program routine representing one or more functions, and can be either connectionless or connection-oriented. A server can exist in any environment that is accessible via the communication network. A connectionless server is not able to establish a session. Its functionality is limited to the execution of a single client request. Connection-oriented servers, in contrast, are able to communicate with their clients to process multiple related requests. Multiple functions that are triggered or selected by a function specification can be integrated into the same server. It is also possible to specify that a request is to be executed synchronously or asynchronously.

Detailed Description Text (14):
Service requests handled by the broker can be divided into three types, as illustrated in FIG. 3B. These request types are remote procedure call ("RPC"), conversational, and message queue.

Detailed Description Text (15):
To the program logic of an application, an RPC is equivalent to a call to a local subroutine--the application executes a call statement, passes control and the appropriate parameters to the called procedure, and waits for control to be passed back from the called procedure. However, unlike a local procedure call, in an RPC the called procedure does not reside within the application (or in the computing machine on which the application is running), but instead is a service provided by a remote participant in the network.

Detailed Description Text (16):
As illustrated in FIG. 5A, the RPC is a synchronous request, in that the client waits for the server to execute the requested remote procedure and return control to the client before the client continues processing. Thus, RPCs do not establish a session with the called procedure--there is no connection or communication between the client and server, and the RPC environment is logically terminated when the request has been processed.

Detailed Description Text (21):
Each participant that requests a service from, or provides a service to, another participant must communicate with the service broker 14. This communication is provided by communication network 22. Communication between devices across a LAN or WAN takes place over a physical medium (wire or fiber-optic cable, satellite or microwave link, etc.), shown in FIG. 3A as physical network 52. The content of the communication is carried by signals that are arranged according to a physical communication protocol 40 (such as SNA (LU 6.2), TCP/IP, DECnet, and LANs). To provide communication between participants and the service broker 14 that is independent of the physical communications protocol 40, one or more higher communication layers are required.

Detailed Description Text (22):
Accordingly, the communications network 22 provides a communication layer between the participants (and broker) and the physical network 52. One example of a product that provides this communication layer is NET-WORK, available commercially from Software, A.G. As shown in FIG. 3A, in this product the communication layer is divided into a network program layer and a network interface program layer 58, with an internal network application program interface ("API") 42 by which the network program communicates with the network interface program 58. The network program layer is further divided into a physical level 54 and a logical level 56. Other products may implement the communication layer differently, as indicated schematically in FIG. 3A by network program 60 and network interface program 62.

Detailed Description Text (23):
The communication layer operates on a protocol referred to herein as the function

server protocol ("FSP"), and presents a low-level application program interface ("LAPI") 44 to the participants and the broker. The communication layer thus provides a common service request format to the participants and then conveys the requests in the appropriate protocol format to the physical network.

Detailed Description Text (39):
The Broker-ID field identifies the broker. It is used when a client requires access to a server registered with a remote broker without directory assistance. This also accommodates the simultaneous use of more than one broker on the same platform.

Detailed Description Text (66):
After the broker receives the request via the client stub, the called procedure in the remote environment is invoked by service dispatcher 18.

Detailed Description Text (69):
An RPC server program can be written in a 3GL as a standard subroutine complying with the standards of the operational environment. This subroutine is dependent on the environment as long as it is implemented in a 3GL. Such servers must be described in the server directory if they are to be known by the broker. When a client request arrives, the broker activates the server in the appropriate environment via the dispatcher. As described above, the subroutine may have a server stub generated by a precompiler so that the server program can receive the remote parameters.

Detailed Description Text (71):
In a programming language that enables the VHAPI, the invocation of a subprogram is identical in both local and remote operations. The language handles all of the logic necessary to establish an RPC transparently. A pseudocode example of a 4GL (NATURAL) call statement is given below in Table 6 for a client and in Table 7 for a server.

Detailed Description Text (75):
G. Network Data Conversion

Detailed Description Text (76):
Different network architectures and hardware platforms have different data representations and parameters. Communications network 22 may therefore perform data conversion from one network architecture/standard to another, such as by translating data between ASCII and EBCDIC formats and performing data compression/decompression and encryption/decryption.

Detailed Description Text (309):
While several brokers can co-exist on the same platform, only one of these can be defined as the "Local" or default broker. All other brokers are "Remote" regardless of the platform on which they are active. External brokers (i.e., brokers that differ from the service broker configuration described herein, can also be enabled to communicate with the broker.

Detailed Description Text (315):
D. Session Manager Gateway

Detailed Description Text (316):
The broker can include a gateway to a session manager, such as the NET-PASS product (commercially available from Software A.G.) to integrate existing IBM 3270-based applications into client/server processing without the need to modify those applications.

Detailed Description Paragraph Table (8):
TABLE 4 _____ CALL "subroutine name" (parameters to be passed to the remote subroutine) _____

CLAIMS:

1. A communication system for managing communication of messages among a plurality of participants in a computing environment, the participants including a server residing on a first computer, and a client residing on a second computer, the first and second computers being heterogeneous, the client and server sending messages to each other, the messages including a request from the client for a service, and the server providing a service, the participants being connected by a physical network with a network protocol, said system comprising:

a protocol independent communications transport layer, said transport layer having a low level application programming interface (LAPI) and accepting messages from the participants via the LAPI and said transport layer communicating with the physical network via the network protocol; and

a service broker, said service broker receiving the service request from the client and determining if the service provided by the server matches the service request, and, if so, communicating the service request to the server.

12. The system of claim 1 wherein the service request is in a higher-level format than the LAPI and further including means for converting the higher-level format service request to the LAPI.

25. A method for managing communication among participants in a computing environment, the participants including a server residing on a first computer and a client residing on a second computer, the first and second computers being heterogeneous, the client and server sending messages to each other, the messages including a request from the client for a service, and the server providing a service, the participants being connected by a physical network with a network protocol, said method comprising the steps of:

disposing between the participants and the physical network a protocol independent communications transport layer having a lower level application interface (LAPI) and communicating with the physical network via the network protocol;

placing a service broker in communication with the participants by said transport layer;

communicating the service request from the client to said service broker;

said service broker determining whether the requested service is provided by the server and, if so;

said service broker communicating the service request from said service broker to the server.

Previous Doc Next Doc Go to Doc#